UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/785,143	02/16/2001	Steven Orodon Hobbs	2007.013000/TDM	4992

22879 7590 07/27/2005

HEWLETT PACKARD COMPANY
P O BOX 272400, 3404 E. HARMONY ROAD
INTELLECTUAL PROPERTY ADMINISTRATION
FORT COLLINS, CO 80527-2400

EXAMINER	
FOWLKES, ANDRE R	

ART UNIT	PAPER NUMBER
2192	

DATE MAILED: 07/27/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/785,143	HOBBS ET AL.	
	Examiner	Art Unit	
	Andre R. Fowlkes	2192	

-- *The MAILING DATE of this communication appears on the cover sheet with the correspondence address* --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *26 April 2005*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-40* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-40* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. This action is in response to the amendment filed on 4/26/05.

Claim Objections

2. The amendment is objected to under 35 U.S.C. 132 and 37 CFR 1.121 as it appears to be introducing new matter not supported by the original disclosure. The original disclosure does not reasonably convey to a designer of ordinary skill in the art that applicant was in possession of the amended subject matter at the time the application was filed. See *In re Rasmussen*, 650 F.2d 1212, 211 USPQ 323 (CCPA 1981).

Specifically, there is no support given, from the original disclosure, for the new limitations of claims 1, 9, 17, 24-26, 33 and 35.

To overcome this objection, applicant may attempt to demonstrate that the original disclosure establishes that he or she was in possession of the amended subject matter or <u>provide the page and line numbers, from the specification, in support of each change in the amended claims</u>.

Claim Rejections - 35 USC § 103

3. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

Claims 1-4, 9-12, 17-19, 24-28 and 33-40 are rejected under 35 U.S.C. 103(a) as being unpatentable over Carr et al. (Carr), "Compiler Optimizations for Improving Data Locality", 1994, ACM, p. 252-262 in view of McGehearty et al., (McGehearty), U.S. Patent No. 6,029,225.

As per claim 1, Carr discloses a method, comprising:

- **identifying a loop** and **each vector memory reference in the loop, in a program** (p. 253 col. L lines 61-62, "data dependence (is determined) between two arrays (vector memory references) ... (in a loop)"),

- **determining dependencies between vector memory references in the loop, including determining unidirectional and circular dependencies** (p. 253 col. L lines 61-62, "data dependence (is determined) between two arrays (vector memory references) ... (in a loop)"),

- **distributing the vector memory references into a plurality of detail loops, wherein the vector memory references that have circular dependencies there between are included in a common detail loop, and wherein the detail loops are ordered according to the unidirectional dependencies between the memory references** (p. 253 col. L lines 2-6, "applying compiler transformations based on data dependence (e.g., loop interchange, fusion, distribution, and tiling) to improve paging... In this paper, we ... integrate optimizations for parallelism and memory", and p. 253 col. L lines 61-62, "data dependence (is determined) between two arrays (vector memory

references) ... (in a loop)", and p. 256 col. L lines 48-51, "Loop distribution separates independent statements in a single loop into multiple loops with identical headers. To maintain the meaning of the original loop, statements in a recurrence (a cycle in the dependence graph) must be placed in the same (common detail) loop (and the detail loops must be ordered according to the unidirectional dependencies between the memory references)").

- analyzing an execution profile of the program after said distributing; and based on the execution profile, determining whether to repeat said identifying a loop, said identifying each vector memory reference, said determining dependencies and said distributing (p. 252 col. L:5-7, "we present compiler optimizations to improve data locality based on a ... cost model (i.e. execution profile)"),

Carr doesn't explicitly disclose **allocating the vector memory references into a plurality of temporary arrays, sized and located, so that none of the vector memory references are cache synonyms**. Additionally, the limitation "so that none of the vector memory references are cache synonyms" is unclear. The examiner is interpreting this limitation as though any one temporary array cannot contain any two vector memory references that are cache synonyms.

However, McGehearty, in an analogous environment, discloses **allocating the vector memory references into a plurality of temporary arrays, sized and located, so that none of the vector memory references are cache synonyms** (col. 2:48-54, "The exact cache collision avoidance mode restructures the loop of moving data to form

a series of loads to get several cache lines staged for loading, each element of data (i.e. vector memory references) is not only moved into the cache, but into registers (i.e. temporary array)... additional loads are interleaved with non-cache conflicting stores (i.e. references that are not cache synonyms) to move new values into memory").

Therefore, it would have been obvious to a person of ordinary skill in the art, at the time the invention was made, to incorporate the teachings of McGehearty into the system of Carr to have **allocating the vector memory references into a plurality of temporary arrays, sized and located, so that none of the vector memory references are cache synonyms**. The modification would have been obvious because one of ordinary skill in the art would have wanted to avoid cache collisions (McGehearty, col. 2:48-57).

As per claim 2, the rejection of claim 1 is incorporated and further, Carr discloses **allocating a plurality of temporary storage areas within a cache and determining the size of each temporary storage area based on the size of the cache and the number of temporary storage areas** (p. 252 col. R lines 9-12, "loop ... distribution ... requires knowledge ... of the cache line size", and p. 252 col. R lines 14-15, "Knowledge of the cache size, associativity, and replacement policy is essential", and the optimization technique of loop distribution includes the allocation of a plurality of temporary storage areas within a cache and determining the size of each temporary storage area based on the size of the cache and the number of temporary storage areas).

As per claim 3, the rejection of claim 1 is incorporated and further, Carr discloses a **section loop including the plurality of detail loops** (p. 256 col. L line 48, "Loop distribution").

As per claim 4, the rejection of claim 1 is incorporated and further, Carr discloses **distributing the vector memory references into a plurality of detail loops further comprises distributing the vector memory references into a plurality of detail loops that each contain at least one vector memory reference that could benefit from cache management** (p. 253 col. L lines 2-6, "applying compiler transformations based on data dependence (e.g., loop interchange, fusion, distribution, and tiling) to improve paging").

As per claim 9, Carr discloses a method, comprising:

- **identifying a loop** and **each vector memory reference in the loop, in a program** (p. 253 col. L lines 61-62, "data dependence (is determined) between two arrays (vector memory references) ... (in a loop)"),

- **determining dependencies between vector memory references in the loop, including determining unidirectional and circular dependencies** (p. 253 col. L lines 61-62, "data dependence (is determined) between two arrays (vector memory references) ... (in a loop)"),

- by distributing the vector memory references into a plurality of detail loops, wherein the vector memory references that have circular dependencies there between are included in a common detail loop, and wherein the detail loops are ordered according to the unidirectional dependencies between the memory references (p. 253 col. L lines 2-6, "applying compiler transformations based on data dependence (e.g., loop interchange, fusion, distribution, and tiling) to improve paging... In this paper, we ... integrate optimizations for parallelism and memory", and p. 253 col. L lines 61-62, "data dependence (is determined) between two arrays (vector memory references) ... (in a loop)", and p. 256 col. L lines 48-51, "Loop distribution separates independent statements in a single loop into multiple loops with identical headers. To maintain the meaning of the original loop, statements in a recurrence (a cycle in the dependence graph) must be placed in the same (common detail) loop (and the detail loops must be ordered according to the unidirectional dependencies between the memory references)"),

- wherein said distributing the vector memory reference into a plurality of detail loops is performed by a first computer for execution by a second computer (p. 256 col. L lines 48-51, "Loop distribution separates independent statements in a single loop into multiple loops with identical headers. To maintain the meaning of the original loop, statements in a recurrence (a cycle in the dependence graph) must be placed in the same (common detail) loop (and the detail loops must be ordered according to the unidirectional dependencies between the memory references)").

Carr doesn't explicitly disclose a plurality of detail loops **that serially proceed through strips of vector memory references and store the strips in temporary arrays so that none of the vector memory references are cache synonyms**.

However, McGehearty, in an analogous environment, discloses a plurality of detail loops **that serially proceed through strips of vector memory references and store the strips in temporary arrays so that none of the vector memory references are cache synonyms** (col. 2:48-54, "The exact cache collision avoidance mode restructures the loop of moving data to form a series of loads (i.e. a serial strip of vector memory reference) to get several cache lines staged for loading, each element of data is not only moved into the cache, but into registers (i.e. temporary array)... additional loads are interleaved with non-cache conflicting stores (i.e. references that are not cache synonyms) to move new values into memory").

Therefore, it would have been obvious to a person of ordinary skill in the art, at the time the invention was made, to incorporate the teachings of McGehearty into the system of Carr to have a plurality of detail loops **that serially proceed through strips of vector memory references and store the strips in temporary arrays so that none of the vector memory references are cache synonyms**. The modification would have been obvious because one of ordinary skill in the art would have wanted to avoid cache collisions (McGehearty, col. 2:48-57).

As per claims 10-12 the Carr/ McGehearty combination also discloses such claimed limitations as addressed in claims 2-4 above, respectively.

As per claim 17 Carr discloses a method, comprising:

- identifying a loop in a program (p. 253 col. L lines 61-62, "data dependence (is determined) between two arrays (vector memory references) ... (in a loop)"),

- identifying each vector memory reference in the loop determining dependencies between vector memory references in the loop(p. 253 col. L lines 61-62, "data dependence (is determined) between two arrays (vector memory references) ... (in a loop)"); **and**

- by distributing the vector memory references into a plurality of detail loops in response to cache behavior and the dependencies between the vector memory references in the loop (p. 256 col. L lines 48-51, "Loop distribution separates independent statements in a single loop into multiple loops with identical headers", and p. 252 col. R lines 14-15, "Knowledge of the cache size, associativity, and replacement policy (i.e. cache behavior) is essential");

- wherein said identifying a loop, said identifying each vector memory reference, said determining dependencies between vector memory references and said distributing the vector memory references into a plurality of detail loops produce code that is substantially independent of a computer architecture; and performing code optimization that are dependant on a computer architecture after said distributing (p. 262 col. L:15-16, "this approach has wide applicability for existing Fortran programs regardless of their original target architecture,").

Carr doesn't explicitly disclose **storage of vector memory references in temporary arrays that are allocated consecutively so that no temporary array elements are cache synonyms**.

However, McGehearty, in an analogous environment, discloses **storage of vector memory references in temporary arrays that are allocated consecutively so that no temporary array elements are cache synonyms** (col. 2:48-54, "The exact cache collision avoidance mode restructures the loop of moving data to form a series of (consecutive) loads to get several cache lines staged for loading, each element of data is not only moved into the cache, but into registers (i.e. temporary arrays)... additional loads are interleaved with non-cache conflicting stores (i.e. references that are not cache synonyms) to move new values into memory").

Therefore, it would have been obvious to a person of ordinary skill in the art, at the time the invention was made, to incorporate the teachings of McGehearty into the system of Carr to have **storage of vector memory references in temporary arrays that are allocated consecutively so that no temporary array elements are cache synonyms**. The modification would have been obvious because one of ordinary skill in the art would have wanted to avoid cache collisions (McGehearty, col. 2:48-57).

As per claim 18, the rejection of claim 17 is incorporated and further, Carr discloses **determining dependencies between vector memory references in the loop, and wherein distributing the loop includes distributing the vector memory references into the plurality of detail loops, wherein the vector memory**

references that have circular dependencies there between are included in a common detail loop (p. 253 col. L lines 61-62, "data dependence (is determined) between two arrays (vector memory references) ... (in a loop)", and p. 256 col. L lines 48-51, "Loop distribution separates independent statements in a single loop into multiple loops with identical headers. To maintain the meaning of the original loop, statements in a recurrence (a cycle in the dependence graph) must be placed in the same loop").

As per claim 19, the rejection of claim 17 is incorporated and further, Carr discloses **determining dependencies between vector memory references in the loop, and wherein distributing the loop includes distributing the vector memory references into the plurality of detail loops, wherein the vector memory references that have circular dependencies there between are included in a common detail loop** (p. 253 col. L line s 61-62, "data dependence (is determined) between two arrays (vector memory references) ... (in a loop)", and p. 256 col. L lines 48-51, "Loop distribution separates independent statements in a single loop into multiple loops with identical headers. To maintain the meaning of the original loop, statements in a recurrence (a cycle in the dependence graph) must be placed in the same loop").

As per claims 24 the Carr/ McGehearty combination also discloses such claimed limitations as addressed in claims 1-3 & 9-12 above.

As per claims 25 the Carr/ McGehearty combination also discloses such claimed limitations as addressed in claims 1-3 & 9-12 above. Additionally, Carr discloses **generating an expanded code of the program;** and **that the expanded code is substantially independent of computer architectures** (p. 262 col. L:15-16, "this approach has wide applicability for existing Fortran programs regardless of their original target architecture").

As per claims 26 the Carr/ McGehearty combination also discloses such claimed limitations as addressed in claims 1-3 & 9-12 above. Additionally, Carr discloses:

- means for determining an execution profile of the program after said distributing occurs (p. 252 col. L:5-7, "we present compiler optimizations to improve data locality based on a ... cost model (i.e. execution profile)"), and

- means for selectively repeating use of said means for identifying a loop, means for identifying each vector memory reference , said means for determining dependencies, said means for distributing the vector memory reference into s plurality of detail loops, and said means for determining an execution profile based on said execution profile (p. 253 col. L lines 2-6, "applying compiler transformations based on data dependence (e.g., loop interchange, fusion, distribution, and tiling) to improve paging... In this paper, we ... integrate optimizations for parallelism and memory", and p. 253 col. L lines 61-62, "data dependence (is determined) between two arrays (vector memory references) ... (in a loop)", and p. 256 col. L lines 48-51, "Loop distribution separates independent statements in a single loop

into multiple loops with identical headers. To maintain the meaning of the original loop, statements in a recurrence (a cycle in the dependence graph) must be placed in the same (common detail) loop (and the detail loops must be ordered according to the unidirectional dependencies between the memory references)").

As per claims 27-28 the Carr/ McGehearty combination also discloses such claimed limitations as addressed in claims 1-3 & 9-12 above.

As per claim 33 Carr discloses a **method for reducing the likelihood of cache thrashing by software to be executed on a computer system having a cache** (p. 252 col. R lines 8-9, "Improve the order of memory accesses to exploit all levels of the memory hierarchy", and exploiting the cache portion of the memory hierarchy, is to use it efficiently, in its designed manner. A cache, operating efficiently in its designed manner of operation, is kept full of the most used memory access locations while cache thrashing is minimized, and p. 252 col. R lines 32-34, "We use the model to derive a loop structure which results in the fewest accesses to main memory (i.e. making the code access the cache and main memory in an efficient manner, thereby reducing cache thrashing)"), **comprising: executing the software on the computer system; generating a profile indicating the manner in which the software uses the cache; identifying a portion of the software using the profile data that may experience cache thrashing; and modifying the identified portion of the software to reduce the likelihood of cache thrashing** (p. 253 col. L lines 61-62, "data dependence (i.e. a

portion of the software that may experience cache thrashing) (is identified) between two arrays ... (in a loop)", and p. 253 col. L lines 2-6, "compiler transformations (are applied) based on data dependence (e.g., loop interchange, fusion, distribution, and tiling) to improve paging (to reduce the likelihood of cache thrashing)... In this paper, we ... integrate optimizations for parallelism and memory").

- **wherein said modifying occurs before optimizations that are based on an architecture of the computer system** (p. 252 col. L:5-7, "we present compiler optimizations to improve data locality based on a ... cost model (i.e. execution profile)").

Carr doesn't explicitly disclose **distributing the cache synonyms into detail loops configured to allocate the cache synonyms into temporary storage arrays, sized and located to prevent cache thrashing**.

However, McGehearty, in an analogous environment, discloses **distributing the cache synonyms into detail loops configured to allocate the cache synonyms into temporary storage arrays, sized and located to prevent cache thrashing**. (col. 2:48-54, "The exact cache collision avoidance mode restructures the loop of moving data to form a series of loads (i.e. cache synonyms) to get several cache lines staged for loading, each element of data is not only moved into the cache, but into registers (i.e. temporary array)... additional loads are interleaved with non-cache conflicting stores (i.e. references that are not cache synonyms) to move new values into memory").

Therefore, it would have been obvious to a person of ordinary skill in the art, at the time the invention was made, to incorporate the teachings of McGehearty into the

system of Carr to **distribute the cache synonyms into detail loops configured to allocate the cache synonyms into temporary storage arrays, sized and located to prevent cache thrashing**.. The modification would have been obvious because one of ordinary skill in the art would have wanted to avoid cache collisions (McGehearty, col. 2:48-57).

As per claim 34, the rejection of claim 33 is incorporated and further, Carr discloses **modifying the identified portion of the software to reduce the likelihood of cache thrashing further comprises: identifying a loop in the identified portion of the software; identifying each vector memory reference in the identified loop; determining dependencies between the vector memory references in the identified loop of the software, including determining unidirectional and circular dependencies; and distributing the vector memory references into a plurality of detail loops, wherein the vector memory references that have circular dependencies there between are included in a common detail loop, and wherein the detail loops are ordered according to the unidirectional dependencies between the memory references** (p. 253 col. L lines 2-6, "applying compiler transformations based on data dependence (e.g., loop interchange, fusion, distribution, and tiling) to improve paging", and p. 253 col. L lines 61-62, "data dependence (is determined) between two arrays (vector memory references) ... (in a loop)", and p. 256 col. L lines 48-51, "Loop distribution separates independent statements in a single loop into multiple loops with identical headers. To maintain the meaning of the original loop,

statements in a recurrence (a cycle in the dependence graph) must be placed in the same (detail) loop (and the detail loops must be ordered according to the unidirectional dependencies between the memory references)").

As per claim 35, this is another method version of the claimed method discussed above, in claim 33, wherein all claimed limitations have also been addressed above.

As per claim 36, the rejection of claim 24 is incorporated and further Carr doesn't explicitly disclose that **the temporary arrays are allocated consecutively such that no temporary array elements are cache synonyms**.

However, McGehearty, in an analogous environment, discloses that **the temporary arrays are allocated consecutively such that no temporary array elements are cache synonyms** (col. 2:48-54, "The exact cache collision avoidance mode restructures the loop of moving data to form a series of loads to get several cache lines staged for loading, each element of data is not only moved into the cache, but into registers (i.e. temporary array)… additional loads are interleaved with non-cache conflicting stores (i.e. references that are not cache synonyms) to move new values into memory").

Therefore, it would have been obvious to a person of ordinary skill in the art, at the time the invention was made, to incorporate the teachings of McGehearty into the system of Carr so that that **the temporary arrays are allocated consecutively such that no temporary array elements are cache synonyms**. The modification would

have been obvious because one of ordinary skill in the art would have wanted to avoid cache collisions (McGehearty, col. 2:48-57).

As per claim 37, the rejection of claim 24 is incorporated and further Carr doesn't explicitly disclose that **the detail loops are allocated into section loops that cause iterative execution of the detail loops based on a size of the strips**.

However, McGehearty, in an analogous environment, discloses that **the detail loops are allocated into section loops that cause iterative execution of the detail loops based on a size of the strips**. (col. 2:48-54, "The exact cache collision avoidance mode restructures the loop of moving data to form a (ordered) series of loads to get several cache lines staged for loading, each element of data is not only moved into the cache, but into registers (i.e. temporary array)... additional loads are interleaved with non-cache conflicting stores (i.e. references that are not cache synonyms) to move new values into memory").

Therefore, it would have been obvious to a person of ordinary skill in the art, at the time the invention was made, to incorporate the teachings of McGehearty into the system of Carr so that that **the detail loops are allocated into section loops that cause iterative execution of the detail loops based on a size of the strips**. The modification would have been obvious because one of ordinary skill in the art would have wanted to avoid cache collisions (McGehearty, col. 2:48-57).

As per claims 38-40 the Carr/ McGehearty combination also discloses such claimed limitations as addressed in claims 1, 17 & 36 above.

6. Claims 5-8, 13-16, 20-23 and 29-32 are rejected under 35 U.S.C. 103(a) as being unpatentable over Carr et al. (Carr), "Compiler Optimizations for Improving Data Locality", 1994, ACM, p. 252-262, in view of McGehearty et al., (McGehearty), U.S. Patent No. 6,029,225, further in view of Mahadevan et al. (Mahadevan) U.S. Patent No. 5,797,013.

As per claim 5, the rejection of claim 1 is incorporated and further, the Carr/ McGehearty combination doesn't explicitly disclose **inserting cache management instructions into at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory**.

However, Mahadevan, in an analogous environment, discloses **inserting cache management instructions into loops to control movement of data associated with the vector memory reference between a cache and main memory** (col. 6 lines 54-55, "(the compiler can) insert prefetches and effect other optimizations (cache management instructions) into the ... loop code").

Therefore, it would have been obvious to a person of ordinary skill in the art, at the time the invention was made, to incorporate the teachings of Mahadevan into the Carr/ McGehearty combination to have cache management instructions inserted into detail loops to control movement of data associated with the vector memory reference between a cache and main memory. The modification would have been obvious

because one of ordinary skill in the art would want to compile the code using techniques that will maximize the efficiency of the compiled code's cache usage and therefore overall operation.

As per claim 6, the rejection of claim 1 is incorporated and further, the Carr/ McGehearty combination doesn't explicitly disclose **inserting prefetch instructions into at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory.**

However, Mahadevan, in an analogous environment, discloses **inserting prefetch instructions into loops to control movement of data associated with the vector memory reference between a cache and main memory** (col. 6 lines 54-55, "(the compiler can) insert prefetches and effect other optimizations into the ... loop code").

Therefore, it would have been obvious to a person of ordinary skill in the art, at the time the invention was made, to incorporate the teachings of Mahadevan into the Carr/ McGehearty combination to have prefetch instructions inserted into detail loops to control movement of data associated with the vector memory reference between a cache and main memory. The modification would have been obvious because one of ordinary skill in the art would want to compile the code using techniques that will maximize the efficiency of the compiled code's cache usage and therefore overall operation.

As per claim 7, the rejection of claim 1 is incorporated and further, the Carr/ McGehearty combination doesn't explicitly disclose **performing loop unrolling on at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory.**

However, Mahadevan, in an analogous environment, discloses **performing loop unrolling on loops to control movement of data associated with the memory reference between a cache and main memory** (col. 6 line 27, "the compiler unrolls loops").

Therefore, it would have been obvious to a person of ordinary skill in the art, at the time the invention was made, to incorporate the teachings of Mahadevan into the Carr/ McGehearty combination to have loop unrolling performed on at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory. The modification would have been obvious because one of ordinary skill in the art would want optimize the performance of the compiled code.

As per claim 8, the rejection of claim 1 is incorporated and further, the Carr/ McGehearty combination doesn't explicitly disclose **inserting at least one of a prefetch instruction and a cache management instruction into at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory, and performing loop unrolling on**

at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory.

However, Mahadevan, in an analogous environment, discloses **inserting a prefetch instruction and a cache management instruction into loops to control movement of data associated with the memory reference between a cache and main memory, and performing loop unrolling on loops to control movement of data associated with the memory reference between a cache and main memory** (col. 6 lines 54-55, "(the compiler can) insert prefetches and effect other optimizations (cache management instructions) into the … loop code", and col. 6 line 27, "the compiler unrolls loops").

Therefore, it would have been obvious to a person of ordinary skill in the art, at the time the invention was made, to incorporate the teachings of Mahadevan into the Carr/ McGehearty combination to have the insertion of at least one of a prefetch instruction and a cache management instruction into at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory, and performing loop unrolling on at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory The modification would have been obvious because one of ordinary skill in the art would want to gain the performance advantages provided by using these optimization techniques in combination (Mahadevan, col. 6 line 22 – col. 7 line 29).

As per claims 13-16, 20-23 and 29-32, the Carr/McGehearty/Mahadevan combination also discloses such claimed limitations as addressed in claims 5-8 above.

Response to Arguments

8. Applicants arguments have been considered but they are not persuasive.

In the remarks, the applicant has argued substantially that:

1) The Carr/McGehearty/Mahadevan combination does not disclose the new limitations of amended claims 1, 9, 17, 24-26, 33 and 35, at p. 18:11-16:25.

Examiner's response:

1) The Carr/McGehearty/Mahadevan combination does disclose the new limitations of amended claims 1, 9, 17, 24-26, 33 and 35 as addressed above in the art rejection of the amended claims.

Conclusion

9. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within

TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Andre R. Fowlkes whose telephone number is (571) 272-3697. The examiner can normally be reached on Monday - Friday, 8:00am-4:30pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on (571)272-3695. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

TUAN DAM
SUPERVISORY PATENT EXAMINER

ARF